USAID | GEORGIA
FROM THE AMERICAN PEOPLE

DATA WAREHOUSE DEVELOPMENT PRIMER – MAY 2012

GEORGIA EDUCATION MANAGEMENT PROJECT (EMP)
SHORT-TERM TECHNICAL ASSISTANCE REPORT

Contract No. AID-114-C-09-00001

11 May 2012

This publication was produced for review by the United States Agency for International Development. It was prepared by J.A. Rodriguez Inc. for Chemonics International Inc. The author’s views expressed in this publication do not necessarily reflect the views of the United States Agency for International Development or the United States Government.

Contents

Acronyms

EMIS	Education Management Information System
EMP	Education Management Project
ERC	Education Resource Center
GB	Gigabyte
GIS	Geographic Information System
IRM	Information Resources Management
MoES	Ministry of Education and Science
MOU	Memorandum of Understanding
OLTP	On Line Transaction Processing
RAM	Random Access Memory
SIS	Student Information System
SAN	Storage Area Network
SQL	Structured Query Language
TB	Terabyte
TOR	Terms of Reference

Introduction

This document should be viewed as a tutorial document for MoES staff to use as a primer to quickly understand how it should go about the process of creating and maintaining the data warehouse that is being developed.

Creating a data warehouse is a significant project with a number of steps. The topics in this document address these steps. They are:

- **Designing a Data Warehouse**
- **Creating the Data Preparation Area**
- **Creating the Data Warehouse Database**
- **Extracting Data from Operational Systems**
- **Cleansing and Transforming Data**
- **Loading Data into the Data Warehouse Database**
- **Preparing Presentation Information**
- **Using a Data Warehouse**
- **Maintaining a Data Warehouse**

Designing a Data Warehouse

The MoES has decided to build a data warehouse instead of an online transaction processing (OLTP) system due to the nature of how it intends to use the data. Designing a data warehouse is very different from designing OLTP system. In contrast to an OLTP system in which the purpose is to capture high rates of data changes and additions, the purpose of a data warehouse is to organize large amounts of stable data for ease of analysis and retrieval. Because of these differing purposes, there are many considerations in data warehouse design that differ from OLTP database design.

Data warehouse data must be organized to meet the purpose of the data warehouse, which is rapid access to information for analysis and reporting. Dimensional modeling is used in the design of data warehouse databases to organize the data for efficiency of queries that are intended to analyze and summarize large volumes of data. The data warehouse schema is almost always very different and much simpler than the schema of an OLTP system designed using entity-relation modeling.

Normally, verification tables used in OLTP systems to validate data entry transactions are not necessary in the data warehouse database. This is because the data warehouse data has been cleansed and verified before it is posted to the data warehouse database, and historical data is not expected to change frequently once it is in the data warehouse. The MoES will need to decide if verification tables will be necessary if the systems that will be providing data to the warehouse do not themselves verify the validity of the data.

Transaction locking considerations, and transactions themselves, play very small roles in data warehouse databases. OLTP systems specialize in large volumes of data update transactions. In contrast, data warehouses specialize in rapid retrieval of information from stable data, and data updates consist primarily of periodic additions of new data.

Backup and restore strategies also differ in a data warehouse from those necessary for an OLTP system. Much of the data in a data warehouse is unchanging history and does not need repetitive backup. Backup of new data can be accomplished at the time of update, and in some situations it is feasible to do these backups from the data preparation database to minimize performance impact on the data warehouse database. Restore policies for a data warehouse might also differ from those for an OLTP, depending on how critical it is for the MoES to have uninterrupted access to data warehouse data.

Data Mart Design

There are two approaches to creating a data warehouse system for an organization. A central data warehouse can be developed and implemented first with data marts created later, or data marts can be implemented such that they make up the data warehouse when their information is joined. In either approach, design must be centralized so that all of the organization's data warehouse information is consistent and usable. Data marts that adhere to central design specifications produce reports that are consistent even though the data resides in different places. For example, a GIS school ID data mart must use the same numbering schema table arranged in the same way as the SIS data mart or summary information will be inconsistent between the two. Given that the MoES will be using stand-alone databases to feed the data warehouse, it should be clear that having consistency between the differing tables will be paramount.

Using Dimensional Modeling

Entity-relation modeling is often used to create a single complex model of all of the organization's processes. This approach has proven effective in creating efficient OLTP systems. In contrast, dimensional modeling creates individual models to address discrete business processes. For example, student information may go to one model, employee to another and financial accounts to yet another. Each model captures facts in a fact table and attributes of those facts in dimension tables linked to the fact table. The schemas produced by these arrangements are called star or snowflake schemas, and have been proven effective in data warehouse design.

Dimensional modeling organizes information into structures that often correspond to the way analysts want to query data warehouse data. For example, the question, "What was the number of dropouts in the Tbilisi district in 2010?" represents the use of three dimensions (student, geography, time) to specify the information to be summarized.

A Data Warehouse Model

A simple dimensional model of grading information might include a fact table named **Grades_Fact** that contains one record for each line item of each student, capturing the grade received, the course, the school, and the date received. Each of these categories of information is organized into its own dimension table. Student information is placed in a **Student** dimension table, School information in a **School** dimension table, time and date information in a **Time** dimension table, and Course and Grade information in a **Course_Grade** dimension table.

In a star schema, each dimension table has a single-part primary key that links to one part of the multipart primary key in the fact table. In a snowflake schema, one or more dimension tables are decomposed into multiple tables with the subordinate dimension tables joined to a primary dimension table instead of to the fact table. In most designs, star schemas are preferable to snowflake schemas because they involve fewer joins for information retrieval and are easier to manage. This would be especially true for the MoES given that so many of the EMISinformation systems will operate in a stand-alone fashion.

Fact Tables

Each data warehouse or data mart includes one or more fact tables. Central to a star or snowflake schema, a fact table captures the data that measures the organization's operations. A fact table might contain student related events such as storing attendance transactions. Fact tables usually contain large numbers of rows, sometimes in the millions of records when they contain one or more years of history for a large organization such as the MoES with a school population of over 650,000 students each having attendance taken several times a day throughout the school year.

A key characteristic of a fact table is that it contains numerical data (facts) that can be summarized to provide information about the history of the operation of the organization. Each fact table also includes a multipart index that contains as foreign keys the primary keys of related dimension tables, which contain the attributes of the fact records. Fact tables should not contain descriptive information or any data other than the numerical measurement fields and the index fields that relate the facts to corresponding entries in the dimension tables.

As an example, one fact table, **attendance_fact_2012**, might contain the following columns:

Column	Description
student_id	Foreign key for dimension table **student**.
time_id	Foreign key for dimension table **time_by_day**.
course_id	Foreign key for dimension table **course**.
attendance_id	Foreign key for dimension table **attendance**.
school_id	Foreign key for dimension table **school**.

In this fact table, each entry represents the attendance of a specific student on a specific day at a specific school.

The most useful measures to include in a fact table are numbers that are additive. Additive measures allow summary information to be obtained by adding various quantities of the measure, such as the attendance rate, by grade, at each school, over a particular time period.

Aggregation in Fact Tables

Aggregation is the process of calculating summary data from detail records. It is often tempting to reduce the size of fact tables by aggregating data into summary records when the fact table is created. However, when data is summarized in the fact table, detailed information is no longer directly available for analysis. If detailed information is needed, the detail rows that were summarized will have to be identified and located, possibly in the source system that provided the data, such as the eStudents database. Fact table data should be maintained at the finest granularity possible. Aggregating data in the fact table should only be done after considering the consequences.Mixing aggregated and detailed data in a fact table can cause issues and complications when using the data warehouse and as such the ministry should refrain from doing this.

Aggregation Tables

Aggregation tables are tables that contain summaries of fact table information. These tables are used to improve query performance when SQL is used as the query mechanism. OLAP technology, such as that provided by Microsoft SQL Server Analysis Services, eliminates the need for such tables. Analysis Services creates OLAP cubes that contain pre-aggregated summaries so that queries can be answered quickly, regardless of the level of summarization required to answer the query. It is not necessary to create aggregation tables in the data warehouse when Analysis Services is used to provide presentation services. Analysis Services creates aggregations as necessary and stores them in tables in the data warehouse database or in internal multidimensional structures. The MoES will need to take this into consideration when developing the reporting system.

Dimension Tables

Dimension tables contain attributes that describe fact records in the fact table. Some of these attributes provide descriptive information; others are used to specify how fact table data should be summarized to provide useful information for reporting purposes. Dimension tables contain hierarchies of attributes that aid in summarization. For example, a dimension containing student information might often contain a hierarchy that separates students into categories such as grade, sex of student, and age of student.

Dimensional modeling produces dimension tables in which each table contains fact attributes that are independent of those in other dimensions. For example, a student dimension table contains data about students, anemployee dimension table contains information about employees, and a school dimension table contains information about schools. Queries use attributes in dimensions to specify a view into the fact information. For example, a query might use the student, school, and time dimensions to ask the question "What was the number of studentsgraduating in the Batumi region in 2012?" Subsequent queries might drill down along one or more dimensions to examine more detailed data, such as "What was the breakdown between males and females?" In these examples, the dimension tables are used to specify how a measure (graduations) in the fact table is to be summarized.

Surrogate Keys

It is important that primary keys of dimension tables remain stable. It is strongly recommended that surrogate keys be created and used for primary keys for all dimension tables. Surrogate keys are keys that are maintained within the data warehouse instead of keys taken from source data systems such as eStudents, SIS, or GIS. There are several reasons for the use of surrogate keys:

- Data tables in various source systems may use different keys for the same entity. Legacy systems that provide historical data might have used a different numbering system than a current processing system. A surrogate key uniquely identifies each entity in the dimension table regardless of its source key. A separate field can be used to contain the key used in the source system.

  Systems developed independently may not use the same keys, or they may use keys that conflict with data in otherpre-existing systems. This situation may not cause problems where each stand-alone system independently reports summary data, but it cannot be permitted in the data warehouse where data is consolidated.

- Keys may change or be reused in the source data systems.
  This situation is usually less likely than others, but some systems have been known to reuse keys belonging to obsolete data. However, the key may still be in use in historical data in the data warehouse, and the same key cannot be used to identify different entities. This would be the case if the ministry changes between different standardized tests.

- Changes in organizational structures may move keys in the hierarchy.
  This can be a common situation. For example, if an employee is transferred from one school to another, the ministry may wish to track two things: courses taught data for the employee prior to the transfer date, and courses taught data for the employee at the new school after the transfer date. To represent this organization of data, the employee's record must exist in two places in the employee dimension table, which is not possible if the employee identification number is used as the primary key for the dimension table. A surrogate key allows the same person to participate in different locations in the dimension hierarchy.

  In this case, the employee will be represented twice in the dimension table with two different surrogate keys. These surrogate keys are used to join the employee’s records to the sets of facts appropriate to the various locations the employee worked at.

  The employee's identification number should be carried in a separate column in the table so information about the employee can be reviewed or summarized regardless of the number of times the employee's record appears in the dimension table.
  Dimensions that exhibit this type of change are called slowly changing dimensions.

  The implementation and management of surrogate keys is the responsibility of the data warehouse. OLTP systems are rarely affected by these situations, and the purpose of these keys is to accurately track history in the data warehouse. Surrogate keys are maintained in the data preparation area during the data transformation process.

Referential Integrity

Referential integrity must be maintained between all dimension tables and the fact table. Each fact record contains foreign keys that relate to primary keys in the dimension tables. Every fact record must have a related record in every dimension table used with that fact table. Missing records in a dimension table can cause facts to be ignored when the dimension table is joined to the fact table to respond to queries or for the population of OLAP cubes. Queries can return inconsistent results if records are missing in one or more dimension tables. Queries that join a defective dimension table to the fact table will exclude facts whereas queries that do not join the defective dimension table will include those facts.

Shared Dimensions

A data warehouse must provide consistent information for similar queries. One method to maintain consistency is to create dimension tables that are shared and used by all components and data marts in the data warehouse. Candidates for shared dimensions include students, school, employees, and geographical dimensions.

Indexes

Indexes play an important role in data warehouse performance, as they do in any relational database. Every dimension table must be indexed on its primary key. Indexes on other columns such as those that identify levels in the hierarchical structure can also be useful in the performance of some specialized queries.

The fact table must be indexed on the composite primary key made up of the foreign keys of the dimension tables.

These are the primary indexes needed for most data warehouse applications because of the simplicity of star and snowflake schemas. Special query and reporting requirements may indicate the need for additional indexes.

Creating the Data Preparation Area

The MoES will need to create tables and other database objects to support the data extraction, cleansing, and transformation operations required to prepare the data for loading into the data warehouse. It can do this by creating a separate database for the data preparation area, or it can create these items in the data warehouse database. The current approach that the ministry is taking is to create the item in the data warehouse database itself.

The data preparation area should include tables to contain the incoming data, tables to aid in implementing surrogate keys, and tables to hold transformed data. Other tables may be required for reconciling data from diverse data sources; such tables may contain cross-reference information to identify common entities such as customer records from systems that use different keys. A variety of temporary tables may also be needed for intermediate transformations.

The specific design of the data preparation area will depend on the diversity of data sources, the degree of transformation necessary to organize the data for data warehouse loading, and the consistency of the incoming data.

Data that is ready to load into the data warehouse should be in tables that have schemas identical to the target tables in the data warehouse. If not, the data should be ready to load into the data warehouse tables through a transformation that can be accomplished in a single step as it is loaded.

The data preparation area should also contain the processes that are used to extract the data from the data sources, the processes that transform and cleanse the data, and the processes that load the data to the data warehouse. These processes may be in the form of SQL queries, stored procedures, Data Transformation Services (DTS) packages, or documents of manual instructions. As in the development of any database system, the objective is to automate as much of the process as possible and to manage and maintain the automated tools developed. Storing and maintaining the transformation processes in the data preparation area permits the use of standard database backup and restore mechanisms to preserve them.

Regardless of whether a separate database is used, creating the data preparation area involves creating tables, views, indexes, DTS packages, and other elements common to relational databases.

Data Preparation Area

Data to be used in the data warehouse must be extracted from the data sources, cleansed and formatted for consistency, and transformed into the data warehouse schema. The data preparation area, sometimes called the data staging area, is a relational database into which data is extracted from the data sources, transformed into common formats, checked for consistency and referential integrity, and made ready for loading into the data warehouse database. The data preparation area and the data warehouse database can be combined in some data warehouse implementations as long as the cleansing and transformation operations do not interfere with the performance or operation of serving the end users of the data warehouse data. Performing the preparation operations in source databases is rarely an option because of the diversity of data sources and the processing load that data preparation can impose on online transaction processing systems.

After the initial load of a data warehouse, the data preparation area is used in an ongoing basis to prepare new data for updating the data warehouse. In most data warehouse systems, these ongoing operations are performed on a periodic basis, often scheduled to minimize performance impact on the operational data source systems. The MoES might wish to consider this on a semester and end of year basis.

The use of a data preparation area that is separated from the data sources and the data warehouse promotes effective data warehouse management. Attempting to transform data in the data source systems can interfere with OLTP performance, and many legacy systems do not have effective or easily implemented transformation capabilities. Reconciliation of inconsistencies in data extracted from various sources can rarely be accomplished until the data is collected in a common database, at which time data integrity errors can more easily be identified and rectified.

The data preparation area should isolate raw data from the data warehouse data to preserve the integrity of the data warehouse and permit it to perform its primary function of preparing information for presentation and supporting access by clients. If the data warehouse database is used for data preparation, care should be taken to avoid introducing errors into the data warehouse data and to minimize the effect of data preparation processing on the performance of the data warehouse. Many data warehouse database operations require sophisticated queries and the processing of large amounts of data; data cleansing can interfere with these operations.

The data preparation area is a relational database that serves as a general work area for the data preparation operations. It will contain tables that relate source data keys to surrogate keys used in the data warehouse, tables of transformation data, and many temporary tables. It will also contain the processes and procedures, such as Data Transformation Services (DTS) packages, that extract data from source data systems.

Creating the Data Warehouse Database

As already noted, the data warehouse database schema is often quite simple compared to those of OLTP databases or the data preparation area. A star schema consists of a single fact table and a number of dimension tables. A snowflake schema adds secondary dimension tables. More complex data warehouses may contain multiple fact tables and a number of dimension tables, some of which are common to all fact tables and others that are specific to a single fact table.

For example, a data warehouse may contain both student information and employee information. Because student data and employee data are different in nature, they should be stored in different fact tables. Some dimension tables, such as a school calendar dimension table, might be common to both students and employees, whereas others might be specific to individual fact tables.

Extracting Data from Operational Systems

Data that will be used in a data warehouse must be extracted from the operational systems that contain the source data. Data is initially extracted during the data warehouse creation, and ongoing periodic extractions occur during updates of the data warehouse. Data extraction can be a simple operation, if the source data resides in a single relational database, or a very complex operation, if the source data resides in multiple heterogeneous operational systems. The goal of the data extraction process is to bring all source data into a common, consistent format so it can be made ready for loading into the data warehouse.

It is better if data in the source operational systems does not contain validation errors. For example, grade records for which there are no corresponding student records to identify the who received the grade are clearly errors in the source data, and should be corrected in the source operational system before the data is extracted for loading into the data warehouse. The MoES may be able to implement error checking in the source operational system so such errors can be detected before extracting data for the data warehouse. If such errors are frequent, the ministry may need to have the operational system examined and modified to reduce such errors because such errors may affect the organization's ability to function as well as its data warehouse.

It is possible the MoES will be able to identify validation errors until the data has been extracted from the operational systems. This situation can occur when data is extracted from multiple data sources. For example, reconciling data extracted from separate systems that relate to student data may uncover discrepancies that must be addressed in one or more of the source systems.

As well, the ministry may also identify inconsistencies other than errors in data after it has been extracted. For example, different data sources may use different coding systems for the same kind of data. You can often use translation tables to reconcile these differences during the extraction operation or later during transformation operations. For example, a legacy system may code school ID’s using a three-character code, whereas another system may use a four-character code. The data from one or both of these systems must be translated into a single set of codes before loading the data into the data warehouse.

In other cases, inconsistencies may be discovered if source systems permit free-form entry of text information. Such data is often internally inconsistent because different data-entry personnel may enter the same data in different ways. Inconsistent representations of the same data must be reconciled if such data is to be used for analysis. For example, in a data source that permits free-form text entry for the school ID portion of an address, the school ID entered as 012 or 12. It may be difficult to modify legacy source systems to implement a standard coding validation. Manual transformation adjustments may be necessary to reconcile such differences if the contributing source systems cannot be modified.

You can use the powerful transformation capabilities of Data Transformation Services (DTS) in Microsoft SQL Serverduring the extraction process to reconcile many formatting, data encoding, and other inconsistencies. Other transformations must be accomplished after the data has been extracted from the source systems.

Some of the tools available in SQL Server for extracting data are:

- Transact-SQL

- Distributed queries
- DTS
- Command line applications
- **bcp** utility
- BULK INSERT statement for loading from text files
- ActiveX scripts

In some data warehouse implementations, you may also find that you can use Replication to copy data from source systems to the data preparation area.

Importing data is the process of retrieving data from sources external to Microsoft SQL Server (for example, a Microsoft Excel file) and inserting it into SQL Server tables. Exporting data is the process of extracting data from an instance of SQL Server into some user-specified format (for example, copying the contents of a SQL Server table to a Microsoft Access database).

Importing data from an external data source into an instance of SQL Server is likely to be the first step performed after setting up the database. After data has been imported into the SQL Server database, one can start to work with the database.

Importing data into an instance of SQL Server can be a one-time occurrence (for example, migrating data from another database system to an instance of SQL Server). After the initial migration is complete, the SQL Server database is used directly for all data-related tasks, rather than the original system. No further data imports are required.

Importing data can also be an ongoing task. For example, a new SQL Server database is created for executive reporting purposes, but the data resides in legacy systems updated from a large number of stand-alone applications. In this case, one can copy new or updated data from the legacy system to an instance of SQL Server on a daily or weekly basis.

Usually, exporting data is a less frequent occurrence. SQL Server provides tools and features that allow applications, such as Access or Microsoft Excel, to connect and manipulate data directly, rather than having to copy all the data from an instance of SQL Server to the tool before manipulating it. However, data may need to be exported from an instance of SQL Server regularly. In this case, the data can be exported to a text file and then read by the application. Alternatively, one can copy data on an ad hoc basis. For example, data can be extracted from an instance of SQL Server into an Excel spreadsheet running on a portable computer and take the computer on a business trip or to be acted on at home.

SQL Server provides tools for importing and exporting data to and from data sources, including text files, ODBC data sources (such as Oracle databases), OLE DB data sources (such as other instances of SQL Server), ASCII text files, and Excel spreadsheets.

Additionally, SQL Server replication allows data to be distributed across an enterprise, copying data between locations and synchronizing changes automatically between different copies of data.

Introducing Replication

Microsoft SQL Server replication is a set of technologies for copying and distributing data and database objects from one database to another and then synchronizing between databases for consistency.

Using replication, one can distribute data to different locations, to remote or mobile users over a local area network, using a dial-up connection, and over the Internet. Replication also allows the ministry to enhance application performance, physically separate data based on how it is used (for example, to separate online transaction processing (OLTP) and decision support systems), or distribute database processing across multiple servers.

Benefits of Replication

Replication offers various benefits depending on the type of replication and the options chosen, but the common benefit of SQL Server replication is the availability of data when and where it is needed.

Other benefits include:

- Allowing multiple sites to keep copies of the same data. This is useful when multiple sites, such as ERC's, need to read the same data or need separate servers for reporting applications.
- Separating OLTP applications from read-intensive applications such as online analytical processing (OLAP) databases, data marts, or data warehouses.
- Allowing greater autonomy. Users can work with copies of data while disconnected and then propagate changes they make to other databases when they are connected.
- Scale out of data to be browsed, such as browsing data using Web-based applications.
- Increasing aggregate read performance.
- Bringing data closer to individuals or groups. This helps to reduce conflicts based on multiple user data modifications and queries because data can be distributed throughout the network, and data can be partitioned based on the needs of different departments or users.
- Using replication as part of a customized standby server strategy. Replication is one choice for standby server strategy. Other choices include log shipping and failover clustering, which provide copies of data in case of server failure.

When to Use Replication

With organizations supporting diverse hardware and software applications in distributed environments, it becomes necessary to store data redundantly. Moreover, different applications have different needs for autonomy and data consistency.

Replication is a solution for a distributed data environment when one needs to:

- Copy and distribute data to one or more sites.
- Distribute copies of data on a scheduled basis.
- Distribute data changes to other servers.
- Allow multiple users and sites to make changes then merge the data modifications together, potentially identifying and resolving conflicts.
- Build data applications that need to be used in online and offline environments.
- Build Web applications where users can browse large volumes of data.
- Optionally make changes at subscribing sites that are transparently under transactional control of the Publisher.

Planning for Replication

Careful planning before replication deployment can maximize data consistency, minimize demands on network resources, and prevent troubleshooting later.

The MoES should consider these areas when planning for replication:

- Whether replicated data needs to be updated, and by whom.
- Data distribution needs regarding consistency, autonomy, and latency.
- The replication environment, including users (especially at remote school sites), technical infrastructure, network and security, and data characteristics.
- Types of replication and replication options.
- Replication topologies and how they align with the types of replication.

Types of Replication

Microsoft SQL Server provides the following types of replication that can be used indistributed applications:

- Snapshot replication
- Transactional replication
- Merge replication

Each type provides different capabilities depending on your application, and different levels of ACID properties (atomicity, consistency, isolation, durability) of transactions and site autonomy. For example, merge replication allows users to work and update data autonomously, although ACID properties are not assured. Instead, when servers are reconnected, all sites in the replication topology converge to the same data values. Transactional replication maintains transactional consistency, but Subscriber (schools or ERCs) sites are not as autonomous as they are in merge replication because Publishers (Central Data Center) and Subscribers generally should be connected continuously for updates to be propagated to Subscribers.

It is possible for the same application to use multiple replication types and options. Some of the data in the application may not require any updates at Subscribers, some sets of data may require updates infrequently, with updates made at only one or a few servers, while other sets of data may need to be updated daily at multiple servers.

Which type of replication is chosen depends on requirements based on distributed data factors, whether or not data will need to be updated at the Subscriber, the replication environment, and the needs and requirements of the data that will be replicated. Each type of replication begins with generating and applying the snapshot at the Subscriber, so it is important to understand snapshot replication in addition to any other type of replication and options the ministry intends to choose.

Replication Tools

Microsoft SQL Server provides several methods for implementing and administering replication, including SQL Server Enterprise Manager, programming interfaces, and other Microsoft Windows components.

SQL Server Enterprise Manager includes a graphical organization of replication objects, several wizards, and dialog boxes you can use to simplify the configuration and administration of replication. SQL Server Enterprise Manager allows one to view and modify the properties of replication configuration, and monitor and troubleshoot replication activity.

The ministry can also implement, monitor, and maintain replication using programming interfaces such as Microsoft ActiveX controls for replication, SQL-DMO, and scripting of Transact-SQL system stored procedures.

Components such as Windows Synchronization Manager and Active DirectoryServices enable the MoES to synchronize data, subscribe to publications, and organize and access replication objects from within Windows applications.

Implementing Replication

The following stages will help the MoES to implement replication, whether using snapshot replication, transactional replication, or merge replication.

Stage	Tasks
Configuring Replication	Identify the Publisher, Distributor, and Subscribers in your topology. Use SQL Server Enterprise Manager, SQL-DMO, or Transact-SQL system stored procedures and scripts to configure the Publisher, create a distribution database, and enable Subscribers.
Publishing Data and Database Objects	Create the publication and define the data and database object articles in the publication, and apply any necessary filters to data that will be published.
Subscribing to Publications	Create push, pull, or anonymous subscriptions to indicate what publications need to be propagated to individual Subscribers and when.
Generating the Initial Snapshot	Indicate where to save snapshot files, whether they are compressed, and scripts to run before or after applying the initial snapshot. Specify to have the Snapshot Agent generate the snapshot one time, or on a recurring schedule.
Applying the Initial Snapshot	Apply the snapshot automatically by synchronizing the subscription using the Distribution Agent or the Merge Agent. The snapshot can be applied from the default snapshot folder or from removable media that can be transported manually to the Subscriber before application of the snapshot.
Synchronizing Data	Synchronizing data occurs when the Snapshot Agent, Distribution Agent, or Merge Agent runs and updates are propagated between Publisher and Subscribers. For snapshot replication, the snapshot will be reapplied at the Subscriber. For transactional replication, the Log Reader Agent will store updates in the distribution database and updates

	will be propagated to Subscribers by the Distribution Agent. If using updatable subscriptions with either snapshot replication or transactional replication, data will be propagated from the Subscriber to the Publisher and to other Subscribers. For merge replication, data is synchronized during the merge process when data changes at all servers are converged and conflicts, if any, are detected and resolved.

Replication Options

Replication options allow the ministry to configure replication in a manner best suited to the application and environment.

Option	Type of Replication	Benefits
Filtering Published Data	Snapshot Replication Transactional Replication Merge Replication	Filters allow you to create vertical and/or horizontal partitions of data that can be published as part of replication. By distributing partitions of data to different Subscribers, one can: ▪ Minimize the amount of data sent over the network. ▪ Reduce the amount of storage space required at the Subscriber. ▪ Customize publications and applications based on individual Subscriber requirements. ▪ Reduce conflicts because the different data partitions can be sent to different Subscribers.
Updatable Subscriptions (Immediate Updating, Queued	Snapshot Replication Transactional Replication	Immediate updating and queued updating options allow users to update data at the Subscriber and either propagate those updates to the Publisher immediately or store the updates in a queue. Updatable subscriptions are best for replication

Updating)		topologies where replicated data is mostly read, and occasionally updated at the Subscriber when Publisher, Distributor, and Subscriber are connected most of the time and when conflicts caused by multiple users updating the same data are infrequent.
Updatable Subscriptions (Merge Replication)	Merge Replication	Merge replication allows users to update data at the Subscriber or Publisher and synchronize changes continuously, on-demand, or at scheduled intervals. Merge replication is well suited for topologies where replicated data is frequently updated at the Subscriber even when the Subscriber is disconnected from the Publisher. Conflicts caused by multiple users updating the same data should be infrequent, but merge replication provides a rich set of options for handling conflicts that do occur.
Transforming Published Data	Snapshot Replication Transactional Replication	One can leverage the data movement, transformation mapping and filtering capabilities of Data Transformation Services (DTS) during replication. With transformable subscriptions, one can: ▪ Create custom partitions for snapshot and transactional publications. ▪ Transform the data as it is being published with data type mappings (for example, integer to real data type), column manipulations (for example, concatenating first name and last name columns into one), string manipulations, and functions.
Alternate Synchronization Partners	Merge Replication	Alternate synchronization partners allow Merge Subscribers to synchronize data with servers other than the Publisher at which the subscription originated. This allows the Subscriber to synchronize data when the original Publisher is unavailable, and is also useful for mobile Subscribers that may have access to a faster or more reliable network

		connection with an alternate server.
Optimizing Synchronization	Merge Replication	By optimizing synchronization during merge replication, you can store more information at the Publisher instead of transferring that information over the network to the Subscriber. This improves synchronization performance over a slow network connection, but requires additional storage at the Publisher.

Replication Data Considerations

Special considerations should be taken when publishing certain data types and properties. This section identifies those data types and properties, and it describes solutions for managing them, including:

- Identity range management. Specifying identity range management can help control how data modifications are made at different Subscribers during merge replication or during snapshot or transactional replication with updatable subscriptions.
- Data types with specific uses. Different data types such as **uniqueidentifier** and **timestamp** have specific uses during replication processing, including conflict resolution when changes to the same data are made at multiple servers.
- NOT FOR REPLICATION. Using the NOT FOR REPLICATION option allows you to implement ranges of identity values in a partitioned environment. .

Administering and Monitoring Replication

Microsoft SQL Server replication provides tools to administer and monitor replication agents, replication alerts, and replication processes, ensuring that replication meets the needs of your organization.

Monitoring replication helps with:

- Setting the agent profiles, schedules, properties, and notifications for replication agents.
- Viewing and troubleshooting agent activity, including verifying when agents last ran, monitoring agent activity, and analyzing replication performance.
- Receiving notification through a replication alert when an event occurs on a replication agent.
- Validating subscriptions to ensure that data values are the same at the Publisher and at Subscribers.

- Reinitializing one or all subscriptions to a publication as needed.
- Managing replication agents from a central location.

Replication and Heterogeneous Data Sources

Microsoft SQL Server offers the ability to replicate data to any heterogeneous data source that provides a 32-bit ODBC or OLE DB driver on Microsoft Windows operating systems. Additionally, SQL Server can receive copies of data replicated from Microsoft Access, Microsoft Exchange, Oracle, DB2 Universal, DB2/MVS, and DB2 AS400, as well as other formats.

Heterogeneous Subscribers

Publishing to heterogeneous data sources allows organizations that have acquired different databases to continue providing SQL Server to individuals, schools, or departments using those databases.

Heterogeneous Publishers

SQL Server can subscribe to snapshot or transactional data replicated from Oracle, DB2, Access, and other data sources. This allows companies that are planning to deploy large databases or a data warehouse with SQL Server, or Internet and intranet applications, to gain access to various sources of data. That data can then be consolidated in SQL Server using replication, and placed into a data mart, data warehouse, or multidimensional database designed for SQL Server Analysis Services.

To implement snapshot or transactional replication published by heterogeneous data sources to SQL Server applications, one would need to configure SQL Server with third-party software or using applications built with SQL-DMO and the Replication Distributor Interface.

Replication Security

Microsoft SQL Server replication uses a combination of security methods to protect the data and business logic in an application.

Security	Description
Role Requirements	By mapping user logins to specific SQL Server roles, SQL Server allows users to perform only those replication and database activities authorized for that role. Replication grants certain permission to the **sysadmin** fixed server role, the **db_owner** fixed database role, the current login, and the **public** role.
Connecting to the Distributor	SQL Server provides a secure administrative link between the Distributor and Publisher. Publishers can be treated as trusted or

	non-trusted.
Snapshot Folder Security	With alternate snapshot locations, one can save snapshot files to a location other than at the Distributor (for example, a network share, an FTP site, or removable media). When saving snapshots, it is important to ensure that replication agents have proper permission to write and read the snapshot files.
Publication Access Lists	Publication access lists (PALs) allow one to determine which logins have access to publications. SQL Server creates the PAL with default logins, but one can add or delete logins from the list.
Agent Login Security	SQL Server requires each user to supply a valid login account to connect to the server. Replication agents are required to use valid logins when connecting to Publishers, Distributors, and Subscribers. However, agents can also use different logins and security modes when connecting to different servers simultaneously.
Password Encryption	Passwords used in SQL Server replication are encrypted automatically for greater security.
Security and Replication Options	Filtering replicated data can be used to increase data security, and there are additional security considerations when using dynamic snapshots, immediate updating, and queued updating.
Security and Replication Over the Internet	Different types of replication over the Internet have different security levels. Additionally, when transferring replication files using FTP sites, precautions must be taken to secure the site and still make it accessible to replication agents.

Enhancing Replication Performance

The MoES can enhance the general performance for all types of replication in an application and on the network by:

- Setting a minimum amount of memory allocated to Microsoft SQL Server.

- Using a separate disk drive for the transaction log for all databases involved in replication.
- Adding memory to servers used in replication.
- Using multiprocessor computers.
- Setting a fixed size for the distribution database.
- Publishing only the amount of data required.
- Running the Snapshot Agent only when necessary and at off-peak times.
- Placing the snapshot folder on a drive not used to store database or log files.
- Using a single snapshot folder per publication.
- Using compressed snapshot files.
- Reducing the distribution frequency when replicating to numerous Subscribers.
- Using of pull or anonymous subscriptions.
- Reduce the verbose level of replication agents to '0' except during initial testing, monitoring, or debugging.
- Run agents continuously instead of on very frequent schedules.
- Using the –UseInprocLoader agent property.

Set a Minimum Amount of Memory Allocated to SQL Server

By default, SQL Server changes its memory requirements dynamically based on available system resources. To avoid low memory availability during replication activities, use the **min server memory** option to set the minimum available memory. If the server is a remote Distributor or a combined Publisher and Distributor, oneshould assign it at least 4gigabytes (GB) of memory.

Use a Separate Disk Drive for All Databases Involved in Replication

This applies to the publication database, the distribution database, and the subscription database. One can decrease the time it takes to write transactions by storing the log files on a disk drive different than the one used to store the database. One can mirror that drive, using a Redundant Array of Inexpensive Disks (RAID)-1, if fault tolerance is required. Use RAID 0 or 0+1 (depending on need for fault tolerance) for other database files. This is a good practice regardless of whether or not replication is being used.

Consider Adding Memory to Servers Used in Replication

If there is need to improve replication performance, consider adding memory to the servers used in replication. For example, if the computer is configured with 16 Gigabytes (GB) of memory, consider increasing the memory to 64GB or more. One can use the **sp_configure** stored procedure to assign additional memory to Microsoft SQL Server.

Use Multiprocessor Computers

SQL Server replication agents can take advantage of additional processors on the server. If there is high CPU usage, consider installing a faster CPU or multiple CPUs (symmetric multiprocessing).

Publish Only the Amount of Data Required

Because replication is easy to set up, there is a tendency to publish more data than is actually required. This can consume additional resources within the distribution databases and snapshot files, and can lower the throughput for required data. Avoid publishing unnecessary tables and consider updating publications less frequently.

Run the Snapshot Agent Only When Necessary and at Off-Peak Times

The Snapshot Agent bulk copies data from the published table on the Publisher to a file in the snapshot folder on the Distributor. In SQL Server, the process of generating a snapshot for transactional replication no longer holds table locks on the published tables. Similarly, for merge replication in SQL Server, concurrency is improved and lock duration is reduced when a snapshot is being generated. Although this reduces the impact on concurrently connected users, generating a snapshot is still a resource intensive process and is best scheduled during off-peak times.

Place the Snapshot Folder on a Drive that Does Not Store Database or Log Files

Similarly, the Snapshot Agent will perform a sequential write of data to the snapshot folder when generating the snapshot for any publication type. Because the snapshot agent always copies a complete copy of the data in the publication to disk when replicating changes, placing the snapshot folder on a separate drive from any database or log files reduces contention among the disks and helps the snapshot process complete faster.

Using a Single Snapshot Folder Per Publication

When specifying the publication properties related to snapshot location, one can choose to generate snapshot files to the default snapshot folder, to an alternate snapshot folder, or to both. Generating snapshot files in both locations requires additional processing when the Snapshot Agent runs. This takes more time than generating the snapshot files to a single location for the publication.

Consider Using Compressed Snapshots

Compressing snapshot files in the alternate snapshot folder can reduce snapshot disk storage requirements and, in some cases, improve the performance of transferring snapshot files across the network when they are used for replication over the Internet. However, compressing the snapshot requires additional processing by the Snapshot Agent when generating the snapshot files, and by the merge agent when applying the snapshot files. This may slow down snapshot generation and increase the time it takes to apply a snapshot in some cases. Consider these tradeoffs carefully when using compressed snapshots.

Reduce the Distribution Frequency When Replicating to Numerous Subscribers

A single Distributor can distribute transactions to a larger number of Subscribers if the Distribution and Merge Agents associated with each Subscriber are scheduled to run less frequently. Stagger when the Distribution Agents or Merge Agents are initially run so they do not all attempt to start simultaneously the first time they are started. If the agents are running on a scheduled basis, the schedules are set by default so that the agents are not running at the same time for regular synchronizations.

Consider Pull or Anonymous Subscriptions

The Distribution or Merge Agent runs on the Distributor for push subscriptions and on Subscribers for pull or anonymous subscriptions. Using pull or anonymous subscriptions can increase performance by moving Distribution or Merge Agent processing from the Distributor to Subscribers.

One can also offload agent processing by using Remote Agent Activation. Agent processing can be moved to the Subscriber for push subscriptions and to the Distributor for pull subscriptions. Administration of the agent still takes place at the Distributor for push subscriptions and at the Subscriber for pull subscriptions. Anonymous subscriptions, which are especially useful for Internet applications, do not require that information about the Subscriber be stored in the distribution database at the Distributor for transactional replication and reduces the storage of information about the Subscriber in the publishing database for merge replication. This reduces the resource demands on the Publisher and Distributor because they do not have to maintain information about anonymous Subscribers.

Anonymous subscriptions are a special category of pull subscriptions. In regular pull subscriptions, the Distribution or Merge Agent runs at the Subscriber (thereby reducing the resource demands on the Distributor), but still stores information at the Publisher. When a publication supports anonymous subscriptions, the publication is configured to always have a snapshot ready for new Subscribers.

For transactional replication, this means that every time the Snapshot Agent runs, a new snapshot will be generated. Typically, a snapshot is not generated if there are no new Subscribers waiting for a snapshot or no Subscriber needs to be reinitialized at the time the Snapshot Agent is run. So while anonymous Subscribers can reduce the resource demands at the Distributor, the tradeoff is that a snapshot is generated more often. With merge replication, a new snapshot is always generated when the Snapshot Agent runs regardless of the type of subscriptions supported by the publication.

Additional Indexes at the Subscriber

If a subscription database needs to be used for decision support analysis and you add a lot of indexes to support these queries, the MoES should note that these additional indexes may significantly reduce the throughput with which changes can be applied to the Subscriber by the Distribution Agent or Merge Agent. In some cases, when aggregating the data at the Subscriber, it may be more efficient to create an indexed view at the Publisher and publish it as a table to the Subscriber using transactional replication.

Application Logic in Triggers at the Subscriber

Similarly, additional business logic in user defined triggers at the Subscriber may also slow down the replication of changes to the Subscriber. For transactional replication, it can be more efficient to include this logic in custom stored procedures used to apply the replicated commands.

Use Horizontal Partitioning Judiciously

When a transactional publication is set up with an article(s) that is horizontally partitioned, the log reader has to apply the filter to each row affected by an update to the table as it scans the transactions log. The throughput of the log reader will therefore be affected. If achieving maximum throughput is key, the ministry should consider using DTS custom partitions to do custom horizontal partitions. That allows the log reader agent to move transactions out of the published database's log as quickly as possible. Instead of affecting all Subscribers with the overhead of filtering the data, only the subscriber that chooses to use a DTS package to filter the data is affected.

Similarly, merge replication must evaluate changed or deleted rows to determine every time changes are synchronized to determine which Subscribers should receive those rows. When horizontal partitioning is employed to reduce the subset of data required at a Subscriber, this processing is more complex and can be slower than when you publish all rows in a table. Consider carefully the tradeoff between reduced storage requirements at each subscriber and the need for achieving maximum throughput.

Use a Fast Network

The propagation of changes to the Subscriber can be significantly enhanced by using a very fast network of 100 Mbps or faster.

Reduce the Verbose Level of Replication Agents

Reduce the **–HistoryVerboseLevel** parameter and/or the **–OutputVerboseLevel** parameter of the Distribution Agents or Merge Agents to the lowest value. This will reduce the amount of new rows inserted to track agent history and output. Instead, previous history messages with the same status will be updated to the new history information. Changing this agent parameter can yield a significant performance gain of up to or over 10 to 15 percent.

However, one should increase the –HistoryVerboseLevel for testing, monitoring, and debugging so that you have as much history information about agent activity as possible.

Run Agents Continuously Instead of on Very Frequent Schedules

Setting the agents to run continuously rather than creating frequent schedules (such as every minute) will improve replication performance. When the Distribution Agent or Merge Agent is set to run continuously, whenever changes occur, they will be immediately propagated to the other servers that are connected in the topology. Because the agent is continuously running, it does not have to start and stop which causes more work for the server where the agent is running.

Consider Using the –UseInprocLoader Agent Property

The **–UseInprocLoader** agent property improves performance of the initial snapshot for snapshot replication, transactional replication, and merge replication.

When you apply this property to either the Distribution Agent (for snapshot replication or transactional replication) or the Merge Agent (for merge replication), the agent will use the in-process BULK INSERT command when applying snapshot files to the Subscriber.

The **–UseInprocLoader** property cannot be used with character mode **bcp**, and it cannot be used by OLE DB or ODBC Subscribers.

Important When using the **–UseInprocLoader** property, the SQL Server account under which the Subscriber is running must have read permissions on the directory where the snapshot .bcp data files are located. When the **–UseInprocLoader** property is not used, the agent (for heterogeneous Subscribers) or the ODBC driver loaded by the agent (for SQL Server Subscribers) reads from the files, so the security context of the Subscriber SQL Server account is not used.

Backing Up and Restoring Replication Databases

In addition to the regular backup and restore guidelines and procedures for Microsoft SQL Server, additional considerations for backing up and restoring the databases are involved in replication.

The considerations for backing up databases used in snapshot replication, transactional replication, or merge replication vary according to the role the server performs in replication and where the failure occurs in the replication topology.

To restore replication, back up some or all of the following regularly:

- Publisher
- Distributor
- Subscriber(s)

The backup strategy will depend on the need for restoring a replicated environment quickly, and on the degree of complexity you can tolerate in your backup plan. The MoES only needs to back up all databases if it wants to restore any replica immediately from backup while minimizing the likelihood of data loss.

Maintaining a regular backup of the Publisher databases, and leveraging the SQL Server replication built-in ability to reinitialize one or more subscriptions on-demand provides a simple recovery strategy. This strategy can be used to support a large enterprise of mobile, occasionally connected Subscribers that otherwise would not typically participate in regular backup management at each node in the topology. One could further limit regular backups to publication databases and rely on SQL Server replication scripting to provide a method for reestablishing replication if the need to restore the entire replication environment occurs.

Another strategy includes backing up only the Publisher and the Distributor as long as the Publisher and Distributor are synchronized. This strategy allows the ministry to restore a replication environment completely. Backing up a Subscriber is optional but can reduce the time it takes to recover from a failure of the Subscriber.

Basic backup plans can result in a longer time to restore the replication environment. If an application requires the need to restore replication immediately, consider more complex backup and recovery strategies described later in this section.

In most situations, the publications and distribution databases should be backed up after adding or changing replication objects such as articles and subscriptions, or after a schema change is made that affects replication. If the distribution database is restored to a version that is before such a change, the publication database will have to be restored to a version before that change as well.

As part of any backup strategy, always keep a current script of the replication settings in a safe location. This should be done in addition to regular backups of the Publisher, Distributor and the Subscribers. In the event of a total server failure or the need to set up a test environment, the MoES can modify the script by changing the server name references and using the script to help recover replication with the previous settings.

The ministry should also script the enabling and disabling of replication. These scripts are part of the backup of the Publisher or Distributor.

Backing Up the Publisher

Publication databases are the primary or central source, of data in a replication topology; therefore, even the most basic recovery plan should include regular backups at the Publisher. Backing up the Publisher requires backing up the publication database regularly on the server where the Publisher is located. Back up the publication database and then make transaction log backups and/or differential database backups. One can also back up the **master** and **msdb** system databases to protect against total loss of the system and not just the publication database. If shipping transaction logs to a warm standby server, back up the **msdb** system database regularly (which is required if log shipping is used).

Backing Up the Distributor

Backing up the Distributor involves backing up the distribution database, the **msdb** database, and the **master** system database. This allows recovery from almost any type of failure without having to re-create publications or reconfigure replication.

Backing up the Distributor preserves the snapshot of the publication as well as the history, error, and replication agent information for applications. It allows one to recover faster in the event of a Publisher or Distributor failure because there is no need to re-establish replication. Particularly for transactional replication, this strategy requires coordination between backing up the publication database and the distribution database. SQL Server handles this coordination automatically. Back up the distribution database, and then make transaction log backups and differential database backups.

Backing Up the Subscriber

A comprehensive backup recovery strategy may rely on re-initialization of subscriptions in the event that recovery is required, or may include regular backups of each subscription database and relevant system databases at the Subscriber. Backing up the Subscriber involves backing up the subscription database and, optionally, the **msdb** and **master** system databases. The **msdb** and **master** databases need to be backed up only if it is a Subscriber that uses pull subscriptions and only if there is a need to be able to restore after a total system loss.

Backup the subscriptions database and then make transaction log backups and incremental database backups.

Note: Backing up each Subscriber is not required to reestablish replication after a failure. Under most circumstances, backing up the Publisher and Distributor regularly should be sufficient. If the cost of reinitializing a Subscriber is significantly greater than the cost of restoring it from a backup, and the complexity of managing backups among the replicas within the enterprise is manageable, you should consider backing up the Subscriber.

Cleansing and Transforming Data

The MoES can accomplish many data transformations during the process of extracting data from the source systems. However, there are often additional tasks to complete before one can load data into the data warehouse. For example, inconsistent data must be reconciled from heterogeneous data sources after extraction and complete other formatting and cleansing tasks. One should also wait until after the extraction process to incorporate surrogate keys. Some transformations that could technically be accomplished during the extraction process may interfere with the performance or operation of the online source system; one should defer these tasks until after extraction is complete.

After extraction from the source systems, the data should reside in a data preparation area where the cleansing and transformations can be completed before the data is loaded into the data warehouse. The data preparation area can be a separate database or separate tables in the data warehouse database. During the cleansing and transformation phase, one can execute procedures to validate and verify data consistency, transform data into common formats, and incorporate surrogate keys.

The ministry may need to perform manual operations to reconcile data inconsistencies or to resolve ambiguous text field entries. Each time a manual operation is required, it should try to identify a way to eliminate the manual step in future data transformation operations. In some cases, it may be able to modify the source data systems to eliminate the cause at the source. In other cases, it may be able to establish an automated process that will set aside unresolved data for later manual exception processing so the bulk of the data can be loaded into the data warehouse without delay for manual intervention.

Some typical data transformations include:

- Combining multiple name fields into one field.
- Breaking down date fields into separate year, month, and day fields.
- Mapping data from one representation to another, such as TRUE to 1 and FALSE to 0 or postal codes from numeric to text.
- Mapping data from multiple representations to a single representation, such as a common format for telephone numbers, or different rating codes to a common "Good, Average, Poor" representation.
- Creating and applying surrogate keys for dimension table records.

Some of the tools available in SQL Serverfor transforming data are:

- Transact-SQL queries
- DTS packages
- Command line applications
- ActiveX scripts

Loading Data into the Data Warehouse Database

After the data has been cleansed and transformed into a structure consistent with the data warehouse requirements, data is ready for loading into the data warehouse. The MoES may make some final transformation during the loading operation, although it should complete any transformations that could identify inconsistencies before the final loading operation.

The initial load of the data warehouse consists of populating the tables in the data warehouse schema and then verifying that the data is ready for use. One can use various methods to load the data warehouse tables, such as:

- Transact-SQL
- DTS
- **bcp** utility

When data is loaded data into the data warehouse, one is populating the tables that will be used by the presentation applications that make the data available to users. Loading data often involves the transfer of large amounts of data from source operational systems, a data preparation area database, or preparation area tables in the data warehouse database. Such operations can impose significant processing loads on the databases involved and should be accomplished during a period of relatively low system use.

After the data has been loaded into the data warehouse database, verify the referential integrity between dimension and fact tables to ensure that all records relate to appropriate records in other tables. One should verify that every record in a fact table relates to a record in each dimension table that will be used with that fact table.

Data integrity in the reverse order is not necessary. That is, it is not necessary for every record in a dimension table to relate to a record in the fact table. For example, dimensions in a student information related data warehouse typically are shared dimensions, which contain the full sets of students, courses, grades, and so on. A fact table may contain records for a specific time period during which some students did not register any absences.

Most queries that retrieve data from the data warehouse use inner joins between the fact and dimension tables. Such queries will ignore facts for which at least one of the joined dimension tables does not contain a matching record, causing retrieved data to be inaccurate and possibly inconsistent among different queries.

If dimension tables are used containing data that does not apply to all facts, one must include a record in the dimension table that can be used to relate to the remaining facts.

To verify referential integrity in a star schema one can use a simple SQL query that counts the rows returned when all appropriate dimension tables are joined to the fact table using inner joins. The number of rows returned by this query should match the number of rows in the fact table. If one is using a snowflake schema, referential integrity should also be verified between dimension tables and the subordinate tables to which they are linked to verify that no records in any table are eliminated by inner joins to subordinate tables. One should perform this verification by starting with the tables at the lowest level of the snowflake dimension and joining them to the tables at the next higher level, continuing until the primary dimension table has been verified. This is an important step because there can be situations in which the dimension may verify correctly against the current fact table, even though some dimension records are missing; these records will be needed when new facts are added.

Preparing Presentation Information

Because access to data warehouse data is often provided through client applications, there are often tasks that must be performed in the data warehouse to prepare the information for presentation to end users. Part of the data warehouse design effort is to identify any special data configuration requirements necessary for these applications and often to configure the applications themselves.

Using a Data Warehouse

The traditional role of a data warehouse is to collect and organize historical business data so it can be analyzed to assist management in making business decisions. Until recently, access to data warehouses was limited to database experts who could create the sophisticated queries necessary to retrieve, summarize, and format information for use by analysts and executive decision makers. As data warehouses become more common and organizations involve lower levels of management in the decision-making process, the need has become greater for direct end-user access to data warehouse data by people with minimal database expertise.

The data warehouse must accommodate the requirements of a continually increasing variety of applications that access data warehouse data. Most applications must be set up and initially configured before they can work effectively with a data warehouse, and this work is often performed or managed by the data warehouse administrator. In some cases the data warehouse must incorporate modifications in order to meet the requirements of a new application.

In addition to end-user applications for data access, other applications continue to be developed that execute within the data warehouse environment to configure and analyze data in new and powerful ways. Such applications require administration and maintenance by the data warehouse administrator.

New uses for data warehouse technology are continually being developed. Real-time data warehouses, once a term with no meaning, are now emerging for use in educational institutions.

What follows describes how access to a data warehouse takes place.

SQL Queries

End users seldom access data warehouse data directly using Structured Query Language (SQL) queries. Analytical SQL queries can be quite complex, requiring database expertise to create correctly. The volume of data in a data warehouse is often so large that sophisticated SQL techniques are needed to achieve useful performance. A SQL query that joins three or four dimension tables to a fact table containing millions of rows and uses aggregate functions such as SUM to summarize and group the results can impose a significant load on any relational database and often yields performance that is not acceptable for online analysis.

OLAP and Data Mining

Online analytical processing (OLAP) is a technology that uses multidimensional data representations, called cubes, to provide rapid access to data warehouse data. Cubes model data in the dimension and fact tables in the data warehouse and provide sophisticated query and analysis capabilities to client applications.

Data mining uses sophisticated algorithms to analyze data and create models that represent information about the data. Data mining models can be used to predict characteristics of new data or to identify groups of data entities that have similar characteristics.

Microsoft SQL Server Analysis Services provides a powerful server and administrative tools to create and manage OLAP data and serve online client applications. Analysis Services also incorporates data mining algorithms that can analyze relational data in the data warehouse database and multidimensional data in cubes.

Cubes and data mining models must be designed, configured, and processed before they can be used by client applications, and they usually require updating when the data warehouse data is updated.

English Query

English Query provides a system for developing client applications that enable end users to access data using English words and phrases. English Query can be used to access data in the data warehouse database or in cubes created by Microsoft SQL Server Analysis Services.

To develop an English Query application, a model must first be created that relates database tables, fields, cubes, and data to English words and phrases. An English Query application can then be generated and incorporated into custom Web or client applications and made available to end users.

Microsoft Office

Data warehouse data in a SQL Server database can be accessed by Microsoft Office components such as Microsoft Excel or Microsoft Access. However, the volume of data in most data warehouses often dictates that special queries or data tables be created and maintained to support the use of these components by end users. Such special queries and tables must be created and maintained as part of the data warehouse.

One exception is the integration of Excel PivotTables with SQL Server Analysis Services. When Analysis Services is used to create and manage OLAP data, end users can easily connect to cubes through an Analysis server and analyze data online or create cubes on their local computer for offline use.

Web Access and Reporting

Web applications that provide end-user access to data warehouse data are popular because the client can use a standard Web browser instead of an application that must be installed, configured, and maintained. Initially limited to simple viewing of data presented on static Web pages, current technology now enables the creation of sophisticated interactive applications that allow users to query and update data in data warehouses and cubes.

Microsoft SQL Server and its components, such as Analysis Services and English Query, offer a number of ways to query and update data over the Web when used with Microsoft Internet Information Services (IIS). SQL Server supports XML functionality for storing, retrieving, and updating information using XML, XML-Data Reduced (XDR) schemas, and XPath queries over HTTP connections. The PivotTable Service component of Analysis Services can be used with IIS to provide Web access to cubes using Multidimensional Expressions (MDX) syntax for querying. English Query applications can be embedded into Active Server Pages (ASP) or COM-based applications to support Web queries in English.

Web data access applications are developed using APIs provided by SQL Server and its components. Web applications can be as simple as displaying a predefined report or executing predefined queries against the data warehouse database or OLAP cubes, or they can be as complex as any directly connected client-server application. The impact of a Web application on data warehouse design or maintenance is determined by the application.

Offline OLAP Cubes

Cubes used in OLAP provide a multidimensional view of data warehouse data that end users find easy to use and explore as they search for answers to business questions. Microsoft SQL Server Analysis Services provides the capability through its PivotTable Service component for client applications to create subsets of data warehouse cubes and save them locally for offline analysis. End-user applications can also use PivotTable Service in an offline mode to create offline cubes directly from relational databases.

Third-party applications and custom applications can use PivotTable Service to create and manage offline cubes. One end-user application that provides offline cube support is Microsoft Excel.

Offline cubes are created and managed by end-user applications and generally have little impact on data warehouse or cube design. Maintenance of offline cube data is the responsibility of the end user, who can refresh data from online cubes or update offline cubes created from local databases as necessary. Offline cubes do not interfere with normal data warehouse and cube management and maintenance.

Third-Party Applications

Many applications have been commercially developed for use with data warehouses and OLAP cubes. Each application has unique requirements that may or may not require design changes to a data warehouse for effective operation of the application. Some applications operate on the data warehouse to provide additional analysis, management, or maintenance capabilities. Others are client applications that provide analysis capabilities for end users. Commercial applications usually require setup and configuration before they can use data warehouse data effectively. Applications may also need configuration adjustments in order to accommodate changes in the data warehouse and updates to data.

Custom Applications

SQL Serverand its components provide a rich set of application programming interfaces (APIs) that can be used to develop custom applications to enhance and automate data warehouse administration, or to create client applications tailored to the organization's needs.

Maintaining a Data Warehouse

Data warehouses collect and organize historical business data so it can be analyzed to assist organizations in making decisions. To achieve this purpose, the data warehouse is created and initially loaded with the existing historical data. It is then periodically updated with new data from operational data systems. Much of the effort in data warehouse maintenance is involved with updating the data in the data warehouse, adjusting data presentation applications to incorporate new data, and updating data marts.

Topics in this section describe common tasks performed to maintain data warehouses.

Updating Data Warehouse Data

Updating data warehouse data includes periodically extracting data from operational systems, cleansing and transforming the data, and loading the new data into the data warehouse. Each data update also includes tasks that must be accomplished to synchronize cubes if they are used for online analytical processing (OLAP), and to update any data marts that are part of the data warehouse.

The process of extracting, cleansing, and transforming data for a periodic update is essentially the same as the process used in the initial loading of the data warehouse, although the update process is often much less complex and more automated than the initial load process. Procedures and automated tasks developed during the initial load process can reduce the amount of manual effort required during updates. Corrections to source operational systems identified and implemented during the initial load also reduce the number of inconsistencies and errors that must be addressed during updates. However, it is often the case that manual intervention is required during updates to ensure the data is ready for loading into the data warehouse.

One difference between the initial data load and data updates is that verifying the referential integrity should be performed incrementally on update data before it is loaded into the data warehouse and made available to users. Updates often include additions and changes to dimension tables as well as the addition of rows to the fact tables. The new and changed data should be checked for internal consistency as well as verified against existing data in the data warehouse before it is loaded into the data warehouse.

After the update data has been made ready for loading into the data warehouse, one can use Transact-SQL, Data Transformation Services (DTS), or the **bcp** utility to update the data warehouse tables. Depending on the design and implementation of the presentation applications that provide access to data warehouse data for end users, the MoES may need to take the data warehouse offline during the update to prevent inconsistencies in query results.

Administering a Data Warehouse

Administering a data warehouse is both similar to and different from administering an online transaction processing (OLTP) system. It is similar in that data warehouse data is stored and maintained in a relational database, so the tools used to administer relational databases can be used with data warehouses. It is different in that OLTP systems are generally characterized by high-volume transaction updates to volatile data, whereas data warehouses are generally characterized by massive amounts of stable historical data. These differences call for different approaches to data warehouse administrative tasks such as backing up data and automating recurring tasks.

Tuning Data Warehouse Performance

A data warehouse must provide rapid evaluation of queries that analyze and summarize huge numbers of rows of data from multiple joined tables. Microsoft SQL Server provides information you can use to optimize the performance of the relational database that contains the data warehouse data. Database performance can be affected by many choices taken in the logical design of the database, its physical implementation, index tuning, query tuning, and so on.

Although the performance of SQL Server Analysis Services depends to a large extent on the performance of the data warehouse database, its performance is also influenced by the design of the data warehouse database and the Analysis Services cubes. You can also tune the performance of Analysis Services by using tools that analyze usage patterns by adjusting the amount of aggregations that are pre-calculated when cubes are processed, optimizing cube schemas to avoid unnecessary joins, and so on. Computer hardware configurations also affect the performance of Analysis servers.

Conclusion

It is hoped this document will provide needed information and guidance to the MoES as it goes about building out its data warehouse. It should be clear from reading this document that building an appropriate data warehouse will require a great deal of thought and care by the organization in order to implement a system that will grow and adapt over time. Any questions or concerns can be directed to the author at jesse@jesserodriguez.com.